UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Corso di
FONDAMENTI DI PROGRAMMAZIONE

Testing and Debugging

Prof. Marco Mamei

Testing

- Testing is an essential skill for anyone writing software because it gives you confidence that the code is functioning properly.
- Companies take this so seriously that they often write their tests *before* writing their code in a process called test-driven development (TDD).
- We'll learn how to test our code thoroughly and how to help Copilot fix code that doesn't work by modifying our prompts.

- There are two ways that software engineers commonly test their code.
  - The first is called **closed-box testing**, and this approach assumes you know nothing about how the code works. **This kind of testing involves varying the inputs and observing the outputs.**
  - The second approach to testing is called **open-box testing:** we look at the code to see where the errors might occur. The advantage of open-box testing is that by looking at the particular structure of the code, we may see where the code is likely to fail and can design additional tests specific to that code

Closed-box testing	Open-box testing
Requires understanding the function specification to test	Requires both the function specification and the code that implements the function to test
Tests don't require an understanding of what the code does.	Tests should be tailored based on how the code was written.
Testers need not have technical expertise about the code they're testing.	Testers need to be able to understand the code sufficiently well to determine which tests may be more important.
Tests the function by varying inputs and checking against expected results	Can test the function in the same way as closed-box testing but can also have more granular tests within a function

Closed-box Testing

- Let's imagine we're trying to test a function that takes in a list of words (strings) and returns the longest word:

```
def longest_word(words):
```

Shorthand for expressing test cases

When writing tests for a function, the standard format is to write the function name and its input along with the desired outcome. For example, the call

```
>>> longest_word(['a', 'bb', 'ccc'])
'ccc'
```

means that if we call the function `longest_word` with the input list `['a', 'bb', 'ccc']`, then the value returned from the function should be `'ccc'`.

Closed-box Testing

- There are two categories for which we typically think about writing test cases:
  - ***Common use cases***—These cases include some standard inputs you could imagine the function receiving and the corresponding result.

```
>>> longest_word(['cat', 'dog', 'bird'])
'bird'
>>> longest_word(['happy'])
'happy'
```

  - ***Edge cases***—These cases are uncommon but possible cases that might break the code.

```
>>> longest_word(['cat', 'dog', 'me'])
'cat'
>>> longest_word(['', ''])
''
```

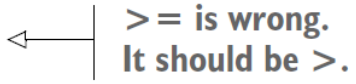
  - **Incorrect *input* testing -** Another category of tests will test the function on how it responds when given incorrect input. A few examples of calling this function with incorrect inputs might be to give the function a nonexisting list by using the value None instead of an actual list (e.g., longest_word(None)), to give the function an empty list (e.g., longest_word([])), to give the function a list with integers as input (e.g., longest_word ([1,2])), or to provide a list of strings but have the strings contain spaces or more than single words (e.g., longest_word(['hi there', ' my ', 'friend'])).

Open-box Testing

- Open-box testing examines the code to see if there are additional kinds of test cases to check.

```
def longest_word(words):
    '''
        words is a list of words

        return the word from the list with the most characters
        if multiple words are the longest, return the first
        such word
    '''
    longest = ""
    for i in range(0,len(words)):
        if len(words[i]) >= len(longest):
            longest = words[i]
    return longest
```



- When reading the if statement, you might notice that it's going to update the longest word in the list of words when the length of the most recent element is greater than *or equal* to the longest word we've seen so far. This is a mistake; it should be >, not >=, but suppose you aren't sure.
  This would motivate you to write a test case like:

```
>>> longest_word(['cat', 'dog', 'me'])
'cat'
```

Come Testare il Codice

- In generale vorremmo testare il codice:
  - In modo ripetibile
  - Senza dover commentare/scommentare parti di test
- **doctest** è un modulo di python che consente di inserire i test all'interno delle docstring delle funzioni.
- **Bonus!** I doctest aiutano copilot a scrivere codice corretto perché offrono esempi di cosa vogliamo ottenere.

```
def longest_word(words):
    '''
        words is a list of words

        return the word from the list with the most characters
        if multiple words are the longest, return the first
        such word

    >>> longest_word(['cat', 'dog', 'bird'])
    'bird'

    >>> longest_word(['happy', 'birthday', 'my', 'cat'])
    'birthday'

    >>> longest_word(['happy'])
    'happy'

    >>> longest_word(['cat', 'dog', 'me'])
    'cat'

    >>> longest_word(['', ''])
    ''
    '''
    longest = ''
    for i in range(0,len(words)):
        if len(words[i]) > len(longest):
            longest = words[i]
    return longest
```

Shows the test cases for doctest

```
import doctest
doctest.testmod(verbose=True)
```

Code (in main) that calls doctest to perform the test

Come Testare il Codice (output)

```
Trying:
    longest_word(['cat', 'dog', 'bird'])
Expecting:
    'bird'
ok
Trying:
    longest_word(['happy', 'birthday', 'my', 'cat'])
Expecting:
    'birthday'
ok
Trying:
    longest_word(['happy'])
Expecting:
    'happy'
ok
Trying:
    longest_word(['cat', 'dog', 'me'])
Expecting:
    'cat'
ok
Trying:
    longest_word(['', ''])
Expecting:
    ''
ok
1 items had no tests:
    __main__
1 items passed all tests:
   5 tests in __main__.longest_word
5 tests in 2 items.
5 passed and 0 failed.
Test passed.
```

First test in longest_word passed

Second test in longest_word passed

Third test in longest_word passed

Fourth test in longest_word passed

Fifth test in longest_word passed

There are no tests in main (outside the function).

longest_word passed all tests.

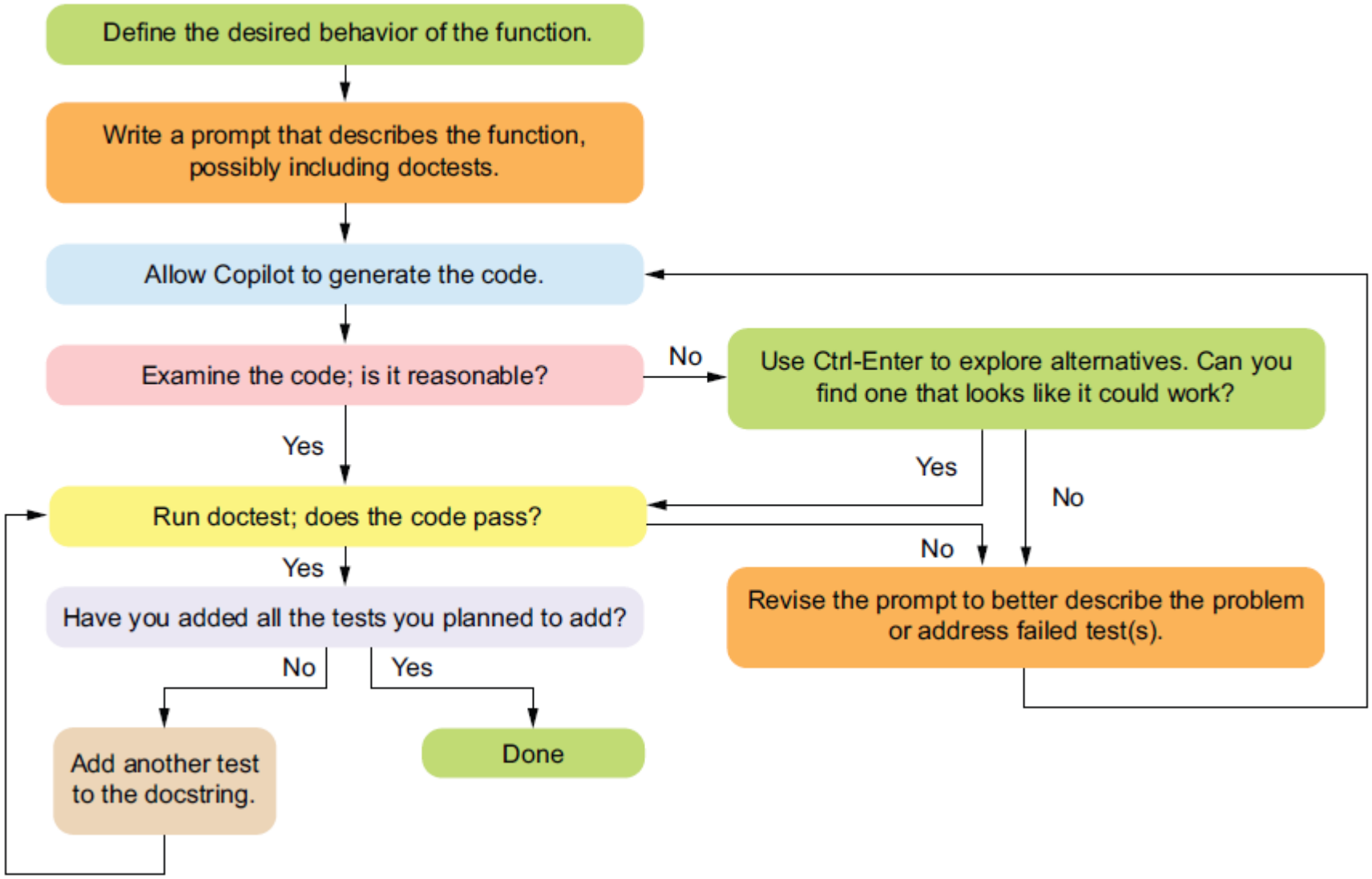
0 failed is what you hope to see.

In realtà ci sono 2 funzioni (items) che stiamo analizzando: __main__ e longest_word.

__main__ non presenta test

longest_word presenta 5 test

Sviluppo Funzioni e test



UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Corso di
FONDAMENTI DI PROGRAMMAZIONE

Debugging

Prof. Marco Mamei

Debugging

- The primary goal of **debugging** is to learn how to find errors (called bugs) in the code and fix them. To find those bugs, you'll gain a deeper understanding of how your code works while you're running it.
  - Syntax errors
  - Logical errors

- Debugging via:
  - Print statements
  - Dedicated tools